Vytvoření HMI pro komunikaci s Nextion displejem a senzory s využitím ESP32

Denis Najman

Předmět: Procedurální programování

January 5, 2025

Abstract

Cílem tohoto projektu bylo navrhnout a implementovat systém pro komunikaci mezi dotykovým resistivním TFT displejem Nextion a mikrokontrolérem ESP32. Tento systém umožňuje komunikaci s PC prostřednictvím sériového portu a slouží jako základ pro budoucí vytvoření HMI (Human-Machine Interface) pro senzory vlhkosti nebo jiných cílových analytů.

Contents

1 Úvod

Cílem tohoto projektu bylo propojit Nextion dotykový displej s mikrokontrolérem ESP32, který slouží jako prostředník pro komunikaci s počítačem. Tento systém poskytuje základ pro další vývoj HMI, které bude schopné komunikovat se senzory vlhkosti nebo jinými plyny, což je konečný cíl tohoto mého zadaného osobního projektu. V rámci implementace byl napsán program v C++/Arduino IDE, který zajišťuje přenos dat mezi ESP32 a Nextion displejem, a zároveň umožňuje komunikaci s PC prostřednictvím sériového portu a možný záznam a zobrazování dat.

Dokument je níže rozdělen na Teoretickou a Praktickou část. Teoretická část obsahuje popis programu tvořeného v Arduino IDE aplikaci a nebyla pro účely tohoto předmětu upravována. Praktická část obsahuje veškerou konanou práci v rámci tohoto předmětu, tedy tvorbu UML diagramu.

2 Popis programu v C++ (Arduino)

Tento program je napsán pro platformu Arduino a jeho hlavním cílem je komunikace mezi mikrokontrolérem ESP32 a Nextion dotykovým TFT displejem. Program čte hodnoty z potenciometru a na jejich základě upravuje hodnoty zobrazené na displeji, a to jak v podobě hodnoty, tak i změny barvy pozadí a textu na displeji. Tento proces zahrnuje také generování sinusového signálu, jehož hodnoty jsou posílány na displej, a komunikaci mezi Arduino a PC pomocí sériového portu (Serial).

2.1 Kód programu

```
1 #include <Arduino.h>
2
3 int potPin = 35;           // Pin pro p ipojen
     potenciometru
4 int delayLength = 200;      // D lka prodlevy mezi
     zm nami
5 long currentMilli = 0;      // Prom nn  pro  asovou
      kontrolu
6 int radianCount = 0;        // Po  tadlo pro
     generov n  sinusov ho sign lu
7 int upCount = 1;            // Po  te n  hodnota
     pro jas
```

```
8  String endChar = String(char(0xff)) + String(char(0
       xff)) + String(char(0xff)); // Konec datov ho
       p  kazu pro Nextion
9  String dataFromDisplay = ""; // Uchov v n
       p ijat ch dat z displeje
10 int lastPotValue = -1;      // Sledov n  posledn
       hodnoty potenciometru
11 int memoryLocation = 10;    // Po  te n  pozice v
       pam ti pro ulo en  dat
12
13 void setup() {
14   Serial.begin(9600);        // Inicializace
         s riov ho portu
15   currentMilli = millis();  // Nastaven
         po  te n ho  asu
16 }
17
18 void loop() {
19   if (millis() > currentMilli + delayLength) { //
         Kontrola, zda uplynul  as  pro zm nu
20     currentMilli = millis();
21     radianCount = radianCount + upCount;    // Zm na
           hodnoty pro generov n  sinusov ho sign lu
22
23     if (radianCount > 63) {    // Omezen
           maxim ln  hodnoty
24       radianCount = 0;
25     }
26
27     float radianFloat = float(radianCount) / 10; //
           P evod na radi ny
28     int sineWaveValue = (sin(radianFloat) * 100) +
           120; // Generov n  hodnoty sinusov ho
           sign lu
29     Serial.print("wepo " + String(sineWaveValue) + ",
           " + String(memoryLocation)); // Odesl n
           hodnoty na displej
30     Serial.write(0xff);
31     Serial.write(0xff);
32     Serial.write(0xff);
33
34     memoryLocation += 4;   // Posun pozice v pam ti
           pro dal   hodnotu
```

```
35      if (memoryLocation > 800) { memoryLocation = 10;
           } // Pokud p es hne hodnotu 800, vr t se
            zp t na 10
36
37      Serial.print("wepo 0," + String(memoryLocation));
             // Odesl n hodnoty do displeje
38      Serial.write(0xff);
39      Serial.write(0xff);
40      Serial.write(0xff);
41    }
42
43    int potValue = analogRead(potPin); // ten
         hodnoty z potenciometru
44    upCount = map(potValue, 0, 1023, 0, 255); //
         Mapov n hodnoty na rozsah pro ovl d n
         jasu
45
46    // Aktualizace hodnoty na slideru, pokud do lo k
         zm n
47    if (potValue != lastPotValue) {
48      Serial.print("add 4,1," + String(upCount)); //
           Odesl n hodnoty na slider
49      Serial.write(0xff); Serial.write(0xff); Serial.
           write(0xff);
50      lastPotValue = potValue; // Aktualizace
           posledn hodnoty potenciometru
51      Serial.print("h0.val=" + String(potValue));
52      Serial.write(0xff); Serial.write(0xff); Serial.
           write(0xff);
53    }
54
55    // ten dat z displeje (pokud je pot eba
         upravit upCount)
56    if (Serial.available()) {
57      dataFromDisplay += char(Serial.read());
58      upCount = dataFromDisplay.toInt();
59      if (dataFromDisplay == "1") {
60        memoryLocation = 10; // Resetov n pam ti
61      }
62      Serial.println(dataFromDisplay);
63      dataFromDisplay = ""; // Vypr zdn n
             etzce pro dal ten
64    }
```

```
65
66    // Změna barvy pozadí a textu na základě hodnoty upCount
67    if (upCount >= 1 && upCount <= 4) {
68      Serial.print("h0.bco=2016");  // Zelená barva pozadí
69      Serial.write(0xff); Serial.write(0xff); Serial.write(0xff);
70      Serial.print("t0.pco=2016");  // Zelená barva textu
71      Serial.write(0xff); Serial.write(0xff); Serial.write(0xff);
72    } else {
73      Serial.print("h0.bco=63488");  //  Červená barva pozadí
74      Serial.write(0xff); Serial.write(0xff); Serial.write(0xff);
75      Serial.print("t0.pco=63488");  //  Červená barva textu
76      Serial.write(0xff); Serial.write(0xff); Serial.write(0xff);
77    }
78  }
```

Listing 1: Program pro komunikaci mezi ESP32 a Nextion displejem

2.2 Popis kódu

2.2.1 Deklarace proměnných

Program začíná deklarací několika proměnných, které slouží k čtení dat z potenciometru, generování sinusového signálu a komunikaci s displejem.

2.2.2 Funkce `setup()`

Funkce `setup()` inicializuje sériovou komunikaci a nastavuje počáteční čas pro kontrolu změn v hlavní smyčce.

2.2.3 Funkce `loop()`

Hlavní smyčka `loop()` provádí:

- Generování sinusového signálu na základě hodnoty `radianCount`.

- Čtení hodnoty z potenciometru a ovládání jasu.
- Posílání dat na displej Nextion.
- Změnu barvy pozadí a textu na displeji podle hodnoty `upCount`.

2.3 Vytvoření GUI pro Nextion displej

GUI pro Nextion displej bylo naprogramováno pomocí aplikace Nextion Editor. Tento editor umožňuje snadné navrhování uživatelského rozhraní, přidávání tlačítek, posuvníků a dalších ovládacích prvků, velmi podobným způsoben jako jsme se učili v Matlabu.

2.4 Komunikace se senzory

`ArduinoProgram` komunikuje se senzorem potenciometru a aktualizuje hodnoty na displeji. Kód posílá hodnoty na displej Nextion, kde jsou vizualizovány pomocí grafických prvků, jako je posuvník.

3 Praktická část

3.1 Příprava prostředí

Pro vytvoření a generování diagramu byla použita aplikace PlantUML a VS Code na systému Windows. Nejdříve bylo nutné stáhnout a nainstalovat následující nástroje:

1. **Java SDK**: Nejprve bylo nutné stáhnout a nainstalovat Javu z oficiálního webu (`https://www.oracle.com/java/technologies/javase-jdk11-downloads.html`).
2. **PlantUML**: Následně byl stažen nástroj PlantUML z `https://plantuml.com/`, který generuje UML diagramy ve formátu png (viz 1) ze souboru s příponou .wsd.
3. **VS Code**: Pro tvorbu PlantUML skriptu (.wsd) bylo použito prostředí VS Code (`https://code.visualstudio.com/`).
4. **PowerShell**: Použití PowerShellu bylo nutné pouze pro ověření instalací a pro spuštění PlantUML skriptu.

3.2 Tvorba UML diagramu

Pro tvorbu diagramu byl vytvořen soubor `.wsd` obsahující strukturu tříd a jejich vztahů. Byly zahrnuty třídy `ArduinoProgram`, `Display`, `Potentiometer` a další. Tyto třídy byly propojeny pomocí vztahů, jako je dědičnost a asociace.

```
@startuml
' Main class
class ArduinoProgram {
    - int potPin
    - int delayLength
    + void setup() : void
    + void loop() : void
}

class Display {
    - int width
    - int height
    + void updateDisplay(message : String) : void
}

class Potentiometer {
    - int value
    + int readValue() : int
}

' Subordinate classes and multilayer
class SerialCommunication {
    + void sendData(String data) : void
    + String receiveData() : String
}

class DisplayElement {
    - String elementId
    - String elementType
    + void updateElement(String data) : void
}

class Sensor {
    - String sensorType
    + int readAnalogValue() : int
}

```

```
class Slider extends DisplayElement {
    + void setPosition(int value) : void
}

class Button extends DisplayElement {
    + void onClick() : void
}

class TemperatureSensor extends Sensor {
    + int readTemperature() : int
}

' Relationships between classes
ArduinoProgram --> Display : "sends data"
ArduinoProgram --> Potentiometer : "reads values"
ArduinoProgram --> SerialCommunication : "manages"
Display --> DisplayElement : "contains"
DisplayElement <|-- Slider
DisplayElement <|-- Button
Potentiometer --> Sensor : "inherits"
Sensor <|-- TemperatureSensor

' Notes
note left of ArduinoProgram
ArduinoProgram is the main program class.
It contains basic loops and initialization.
end note

note right of SerialCommunication
SerialCommunication is responsible
for communication between Arduino and other devices.
end note

note top of Display
The display contains various elements
for data visualization.
end note

note left of TemperatureSensor
TemperatureSensor is an example of a dedicated sensor
,
that extends the basic functionality of the Sensor
   class.
```

```
79 end note
80 @enduml
```

Listing 2: Skript PlantUML pro vygenerování UML diagramu

Diagram zahrnoval poznámky k jednotlivým třídám, které vysvětlují jejich funkce a propojení. Diagram byl následně vygenerován pomocí příkazu:

```
java -jar plantuml-1.2024.8.jar diagram.wsd
```

Vygenerovaný diagram vypadal takto:

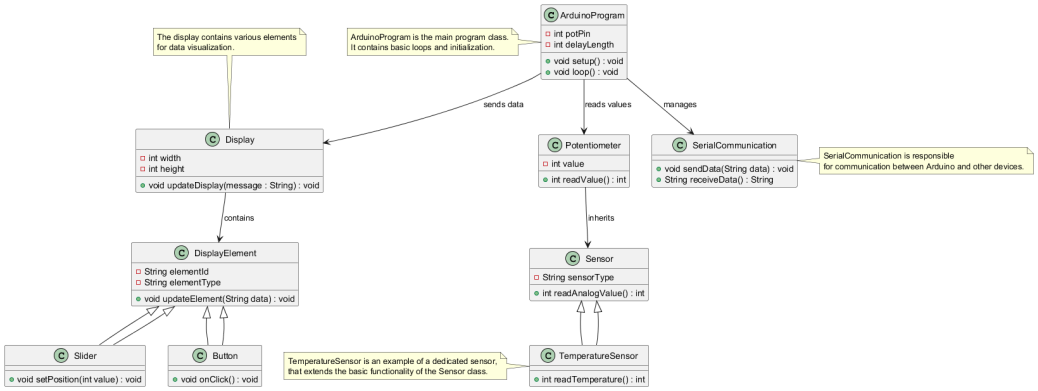

Figure 1: UML diagram Programu v Arduino

4 Závěr

Tento dokument obsahuje dva hlavní skripty: jeden v C++/Arduino IDE, který implementuje komunikaci mezi ESP32 a Nextion displejem, a druhý ve formátu PlantUML, který zobrazuje strukturu tříd a jejich vztahy (class diagram). GUI na Nextion displeji bylo navrženo pomocí aplikace Nextion Editor a není součástí tohoto dokumentu.

Tento projekt slouží jako základ pro vývoj pokročilých HMI (Human-machine interference), které budou schopné zprostředkovat komunikaci člověka s přístroji, jako jsou senzory vlhkosti či senzory jiných kýžených plynů. Další rozvoj tohoto systému bude zaměřen na integraci reálných senzorů a rozšíření funkcionality pro různé aplikace, jako je ovládání teploty senzoru a pozorování příslušných charakteristických odezev senzoru při detekci kýženého plynu.